시작.

1. 목적

ES2015를 활용한 component (module)방식의 웹 프론트엔드 개발.
다른 스텝으로 프레임워크 잘 쓰기. (필요하면 내가 그냥)

2. 순서

ES2015 필수 셋
간단한 UI 완성(AJAX,EVENT, TEMPLATE, DOM)
리팩토링 - Dispatcher기반 MVC로 수정
리팩토링 - webpack활용한 모듈방식의 개발.

ES6.

ES6 === ES2015
(ES2016, ES2017..)

ES2015
개선된 JavaScript문법.
ES6 browser compatibility의 훌륭한 지원.
ES6를 기반으로 한 JavaScript 생태계의 확산.

1. scope enhancements - let

ES6에서는 let 키워드를 사용해서 변수를 선언하면 Block({})단위의 scope를 만들 수 있다.
키워드를 사용하면 됨.

```
var name = 'play ground';
function home() {
  var homeName = 'my house';
  for (let i = 0; i<1000; i++){}
  console.log(i); //i is not defined
}
```

따라서,Block단위로 사용할때는 let을 사용하는 것을 권장.

2. scope enhancements - const(1/2)

const로 선언된 변수는 값을 재 할당 할 수 없다.

```
function home() {
  const homeName = 'my house';
  homeName = 'your house';
}

home() //TypeError: Assignment to constant variable.
```

2. scope enhancements - const(2/2)

주의할점은, const를 사용한다고 수정할수 없음을 의미하는 것은 아니다.
const를 사용하더라도 배열과 오브젝트의 값을 변경하는 것은 가능.

```
function home() {
  const list = ['john', 'adele', 'hary'];
  list.push('tiger');
  return list;
}
home() //["john", "adele", "hary", "tiger"]
```

3. Object enhancements

객체를 쉽게생성

메서드에 function 키워드도 생략가능.

```
const name = "nayoun";
const age = 9;
const others = {
    address : "kwang myeung city",
    tel : null,
    height: 130
}

const data = {
    name,
    age,
    others,
    getName() {
        return this.name;
    }
}
console.log(data.getName());  //nayoun
```

4. Destructuring

Array Destructuring

```
let previousData = ["apple", "orange", 100, 200];
let [,,applecount, orangecount] = previousData;
```

Object Destructuring

```
let obj = {
  name : "crong",
  address : "pororo house",
  age : 12
}

let {name, age} = obj;
console.log(name,age);

//변수 이름을 변경해서 받을 수도 있음.
let {name:myName, age:myAge} = obj;
console.log(myName, myAge);
```

5. Destructuring practice (1/2)

https://gist.github.com/nigayo/787180f0c9756d198df45c2de4fb20db

```
//make title and imgurl array of mbc
let [,mbc] = news;
let {title,imgurl} = mbc;
console.log(title,imgurl);

//또는 이렇게도 가능.
//make title and imgurl array of mbc
let [,{title,imgurl}] = news;
console.log(title,imgurl);
```

6. Destructuring practice (2/2)

https://gist.github.com/nigayo/787180f0c9756d198df45c2de4fb20db

```
//destructuring in function parameters
//let {newslist} = mbc; 와 같이 동작된다고 할 수 있음.
function getNewslist({newslist}) {
  console.log(newslist);
}
getNewslist(mbc);

//make imgurl array.
var urls = news.map(({imgurl}) => imgurl);
console.log(urls);
```

7. template enhancements

```
const data = {
   hour : new Date().getHours(),
   name : "codesquad"
}

const template = `<div><span>hello! ${data.name}</span></(
console.log(template); //<div><span>hello! codesquad</spa
```

8. function enhancements - arrow (1/2)

```
setTimeout(() => {console.log("hello")}, 1000);
setTimeout(() => console.log("hello"), 1000);
```

```
var newArr = [1,2,3].map((v) => {
  return v*2;
});
var newArr = [1,2,3].map((v) => (v*2));
var newArr = [1,2,3].map((v) => v*2);
var newArr = [1,2,3].map(v => v*2);
```

9. function enhancements - arrow (2/2)

주의. arrow를 사용하는 경우, this가 가리키는 부분이 콜백이 실행되는 시점이 아닌 **함수가 정의된 시점의** context 를 기준으로 함.

```
var obj = {
  run() {
        setTimeout(function() {
       console.log(this);
        }, 1000);
  }
}
obj.run(); //window

var obj = {
  run() {
        setTimeout(() => {
       console.log(this);
        }, 1000);
  }
}
obj.run(); //obj
```

10. function enhancements - default parameters

```
function sum(value, count=10, size=20) {
   return value * size;
}

sum(3,10);
```

11. ES6 Class

```
class Health {
  constructor(name, lastTime) {
    this.name = name;
    this.lastTime = lastTime;
  }

  showHealth() {
    console.log("오늘은 " + this.lastTime + "까지 "
     + this.name + " 운동을 하셨네요");
  }
}

var myHealth = new Health("달리기", "23:11");
myHealth.showHealth();
```

12. Module. (import, export)

import와 export를 통해서 모듈을 불러올 수 있다.

```
//calculate.js
export const sqrt = Math.sqrt;
export function square(x) {
    return x * x;
}
export function diag(x, y) {
    return sqrt(square(x) + square(y));
}
```

```
//service.js
import { square, diag } from './calculate.js';
console.log(square(11)); // 121
console.log(diag(4, 3)); // 5
```

```
//main.html
 <script type="module" src='./src/service.js'></script>
```

code : http://2ality.com/2014/09/es6-modules-final.html

13. set

중복없이 유일한 값만 저장됨. 어떤 값이 이미 존재하는지 체크할 때 유용함.

```
let mySet = new Set();

undefined
mySet.add("eagles");
mySet.add("tigers");

Set(2) {"eagles", "tigers"}
mySet.has("eagles");
true
mySet.delete("eagles");
true
mySet.has("eagles");
false
```

여기서부터는 참고하세요

14. Array enhancements - spread operator (1/3)

배열 합치기가 쉽다.

```
let previousData = ["apple", "orange", 100, 200];
let newData = [1,2,3,...previousData];

console.log(newData);
//[1, 2, 3, "apple", "orange", 100, 200]
```

15. Array enhancements - spread operator (2/3)

새로운 배열로 쉽게 복사할 수 있다. (immutable 객체 생성방법)

```
let previousData = ["apple", "orange", 100, 200];
let newData = [...previousData];

console.log(newData === previousData);
//false
```

object 도 spread operator를 사용할 수 있음(bable의 플러그인 필요)

16. Array enhancements - spread operator (3/3)

배열을 function에 개별 파라미터로 전달하기가 쉽다.

```
function sum(a,b) { return a+b}
const arr = [4423,42];

//sum.apply(null, arr);
sum(...arr);
```

17. Array enhancements - methods

from 메서드를 통해서 가짜 배열을 진짜 배열로 취급하기 쉽게 됐다(?)
find 메서드라는 것도 생겼음.

18. Tagged template literals

tagged template은 template 문자열의 파싱이 필요한 경우에 사용할 수 있다.

```
function fn(val, name, hour) {
  var ampm = (hour > 11) ? "pm" : "am";
  console.log(val[0], name, val[1], hour, ampm, val[2]);
}

var data = {
   hour : new Date().getHours(),
   name : "solvin"
}

const template = fn`<div><span>hello! ${data.name},
        current time is ${data.hour}</span></div>`;
```

19. function enhancements - rest parameters

rest parameter를 활용해 임의의 인자를 배열형태로 받을 수 있다.
rest parameter는 진짜 배열임으로 arguments를 사용해야 하는 상황에서는 더 좋다.

```
function checkNumber(...arg) {
  const result = arg.every((v) => typeof v === "number");
  console.log(result);
}

checkNumber(1,2,3,NaN,4,5,null);
```

20. weakSet

참조를 가지고 있는 객체형태만 저장 가능하다.
저장된 객체가 더이상의 참조를 가지지 않을때는 가비지컬렉션의 대상이 된다.

```
let arr = [1,2,3,4];
let arr2 = [...arr,5,6,7,8,9];
let obj = {arr, arr2};

let ws = new WeakSet();
ws.add(arr);ws.add(arr2);ws.add(obj);

console.log(ws);

arr.push(555);
delete obj.arr2;
console.log(ws); //참조가 유지됨으로 변경이 반영됨.

arr2 = null;

console.log(ws.has(arr2); //불필요한 참조는 가비지컬렉션대상으로 취
```

추가활용 참고 : https://esdiscuss.org/topic/actual-weakset-use-cases

21. map and weakmap

set과 달리 키/값 구조로 저장이 된다.

weakMap의경우의 키값은 weakSet과 같이 객체만 가능하다.

```
let wm = new WeakMap();
let fun = function() {};
wm.set(fun,0);

let count = 0;

for(let i=0;i<10;i++){
  count = wm.get(fun);
  count++;
  wm.set(fun,count);
}

console.log(wm.get(fun)); //10

fun = null;
console.log(wm.has(fun)); //false
```

22. WeakMap을 이용한 클래스 생성.

private 변수 만들기. 객체가 필요없어질때는 역시 가비지컬렉션 대상이 됨.

```
const wm = new WeakMap();
class Rectangle {
  constructor(height, width) {
    wm.set(this, {height,width});
  }
  get area() {
    return this.calcArea();
  }
  calcArea() {
    const {height, width, size} = wm.get(this);
    return height * width;
  }
}

const square = new Rectangle(10, 10);
const square2 = new Rectangle(10, 20);
console.log(square.area, square2.area);
```

클래스 참고 :

https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Classes

23.prototype 객체 - setPrototypeOf

Object.create 말고도 prototype객체에 넣는 방법이 ES6에 추가됨.

```
var healthObj = {
  showHealth : function() {
    console.log("오늘은 " + this.lastTime + "까지 "
                + this.name + " 운동을 하셨네요");
  }
}

var myHealth = {
  name : "달리기",
  lastTime : "23:10"
}

Object.setPrototypeOf(myHealth, healthObj);

console.log(myHealth);
```

24. prototype 객체 - Object.assign()

```
var healthObj = {
  showHealth : function() {
    console.log("오늘은 " + this.lastTime + "까지 "
                + this.name + " 운동을 하셨네요");
  }
}

var myHealth = Object.assign(Object.create(healthObj), {
     name : "달리기",
     lastTime : "23:10"
});
```

Others..

- Promise
- Proxy